Automated deployment and customization of routing overlays on PlanetLab

C. Freire, A. Quereilhac,
T. Turletti, W. Dabbous

{claudio-daniel.freire,alina.quereilhac,thierry.turletti,walid.dabbous}@inria.fr

A primer to PlanetLab

- A **worldwide distributed** testbed connected to the Internet
- **1123** nodes worldwide
  - 306 in Europe
  - Not all responsive
- **Constantly changing**
  - Nodes come on and offline, without warning.
- **V-server** virtual hosts
  - Limited access to kernel interfaces
  - Nodes are PCs
  - Slices are groups of resources (nodes) assigned to an experiment
  - Slivers are PC resources (a V-server in a node) assigned to an experiment (slice)

Overlays in PlanetLab

Typical PlanetLab documentation on tunnel creation

Build example **tuntest.c**, which:

- Opens a PL-specific *vsys* socket, and sends a control packet
- Receives a file descriptor associated to the tunnel

Send configuration instructions to another PL-specific *vsys* pipe:

```
cat /vsys/vif_up.out&

[0] cat>  /vsys/vif_up.in
      <name of interface, i.e. [tun|tap]<sliceid>-n
      <ip address e.g. 172.16.2.1>
      <netmask e.g. 24>
      <options as newline-separated name-value pairs>
      <Control-D>
```

Any task needs a customized program to be written

Overlays in PlanetLab

- Too much work even for the simplest experiments
- Very low re-usability (cannot share easily)
- Encourages repeating the same mistakes

Overlays in PlanetLab

- Too much work even for the simplest experiments
- Very low re-usability (cannot share easily)
- Encourages repeating the same mistakes

Solution: Automate

- Make simple experiments simple
- Encourage re-usability
  - share solutions to common mistakes
- Allow customization
  - because it's needed every time
    - Custom aggregation methods
    - Custom encapsulation methods
    - Custom routing protocols

Some existing tools

- VINI/IIAS
- RiaS
- Trellis
- Splay
- Plush/Gush

Some existing tools

- VINI/IIAS
  - UML provides complete virtualization while sacrificing performance
  - Difficult customization of encapsulation methods
    - Requires modifying PL-VINI's version of the Click router
    - Precludes usage of pre-existing prototypes
- RiaS
- Trellis
- Splay
- Plush/Gush

Some existing tools

- VINI/IIAS
- RiaS
  - Runs on any PlanetLab node
  - Exclusively user-mode packet forwarding
  - No customization support
  - It requires "router" nodes to run inside PL-VINI
- Trellis
- Splay
- Plush/Gush

Some existing tools

- VINI/IIAS
- RiaS
- Trellis
  - GRE tunnels with kernel-mode forwarding
  - Software in-kernel switches route layer-2 packets
  - No layer-3 routing support
  - No automation
  - Cannot scale to widespread adoption
- Splay
- Plush/Gush

Some existing tools

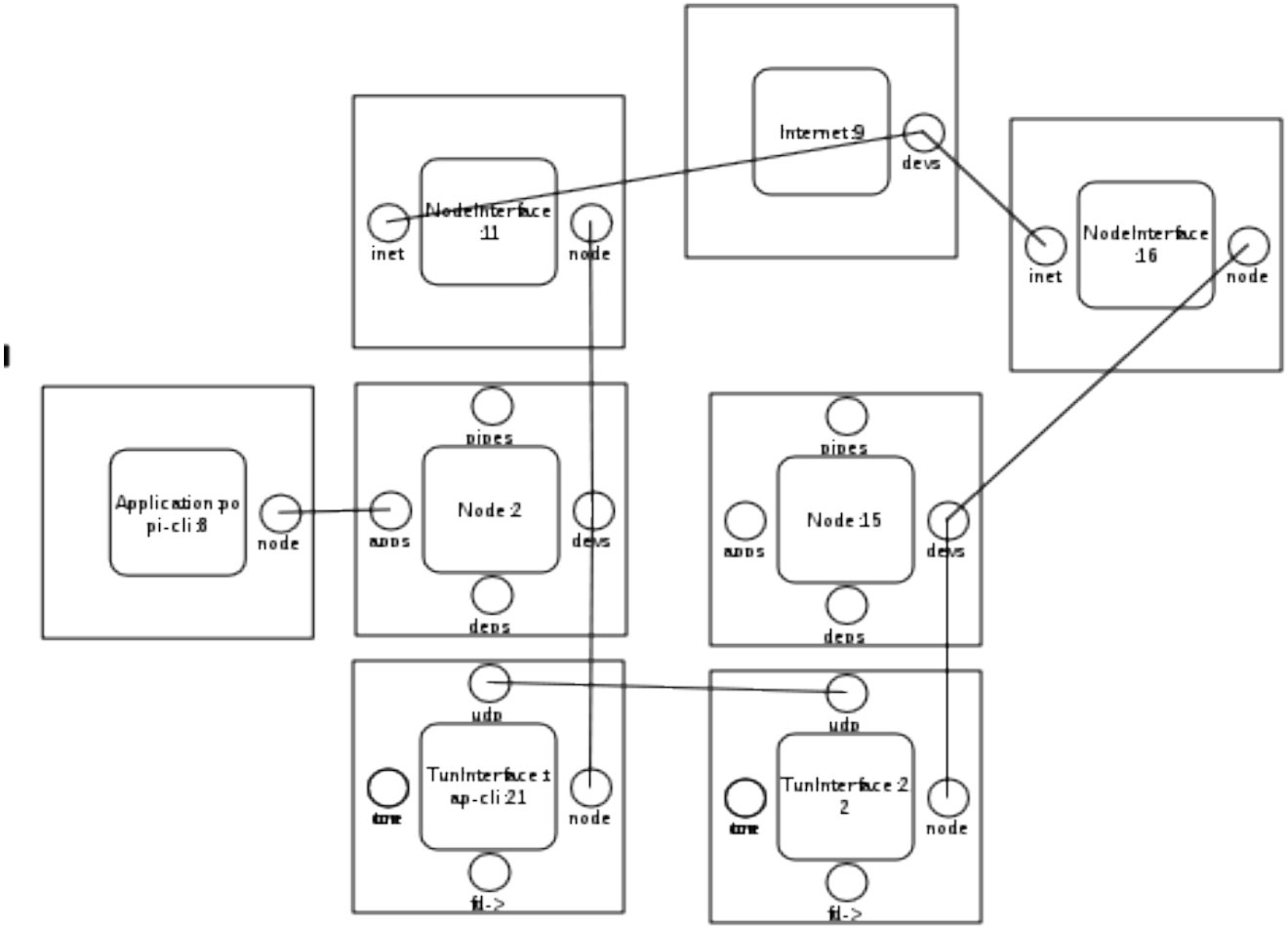
- VINI/IIAS
- RiaS
- Trellis
- Splay
  - Fully automated deployment and trace collection
  - Exclusively application-level
    - Even packet loss is emulated at the application level
  - Only supports Ruby applications
- Plush/Gush

Some existing tools

- VINI/IIAS
- RiaS
- Trellis
- Splay
- Plush/Gush
  - Fully automated deployment and monitoring
  - Accepts any kind of application
  - Lacks high-level abstractions for overlay construction
    - Does not solve routing or tunneling problems
    - Though it does allow users to implement their own solution

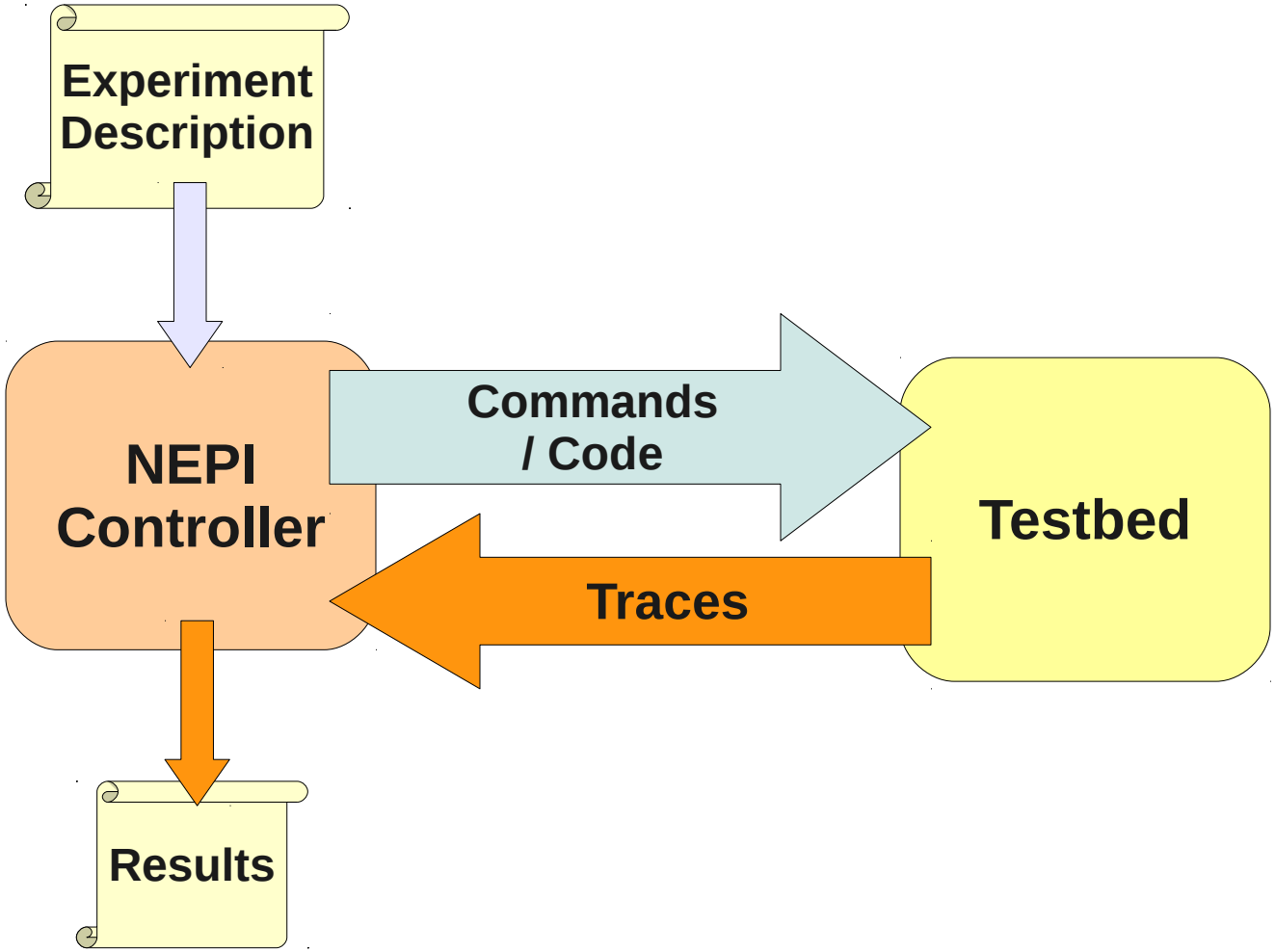
NEPI

Experiment description



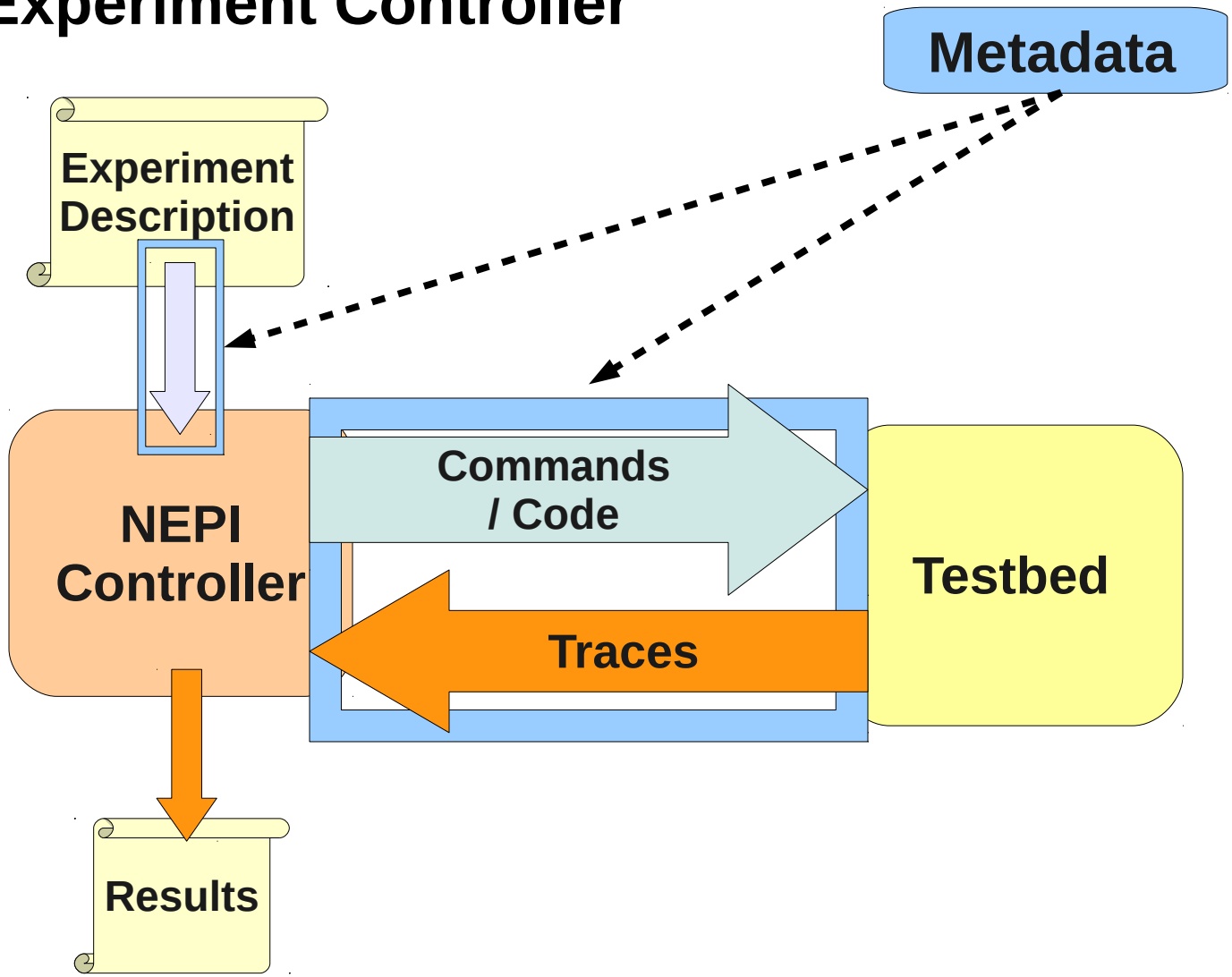
NEPI

Experiment Controller



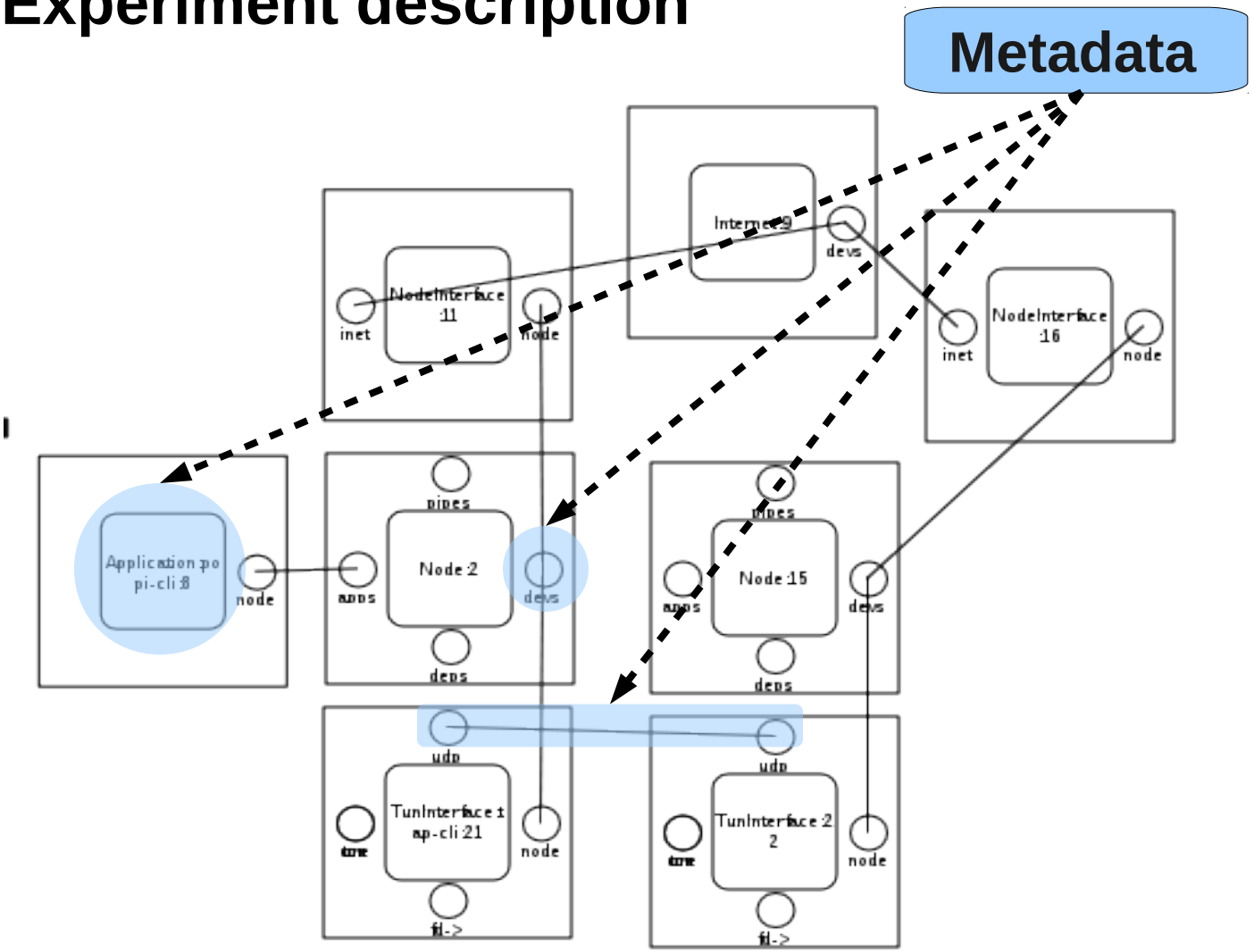
NEPI

Experiment Controller



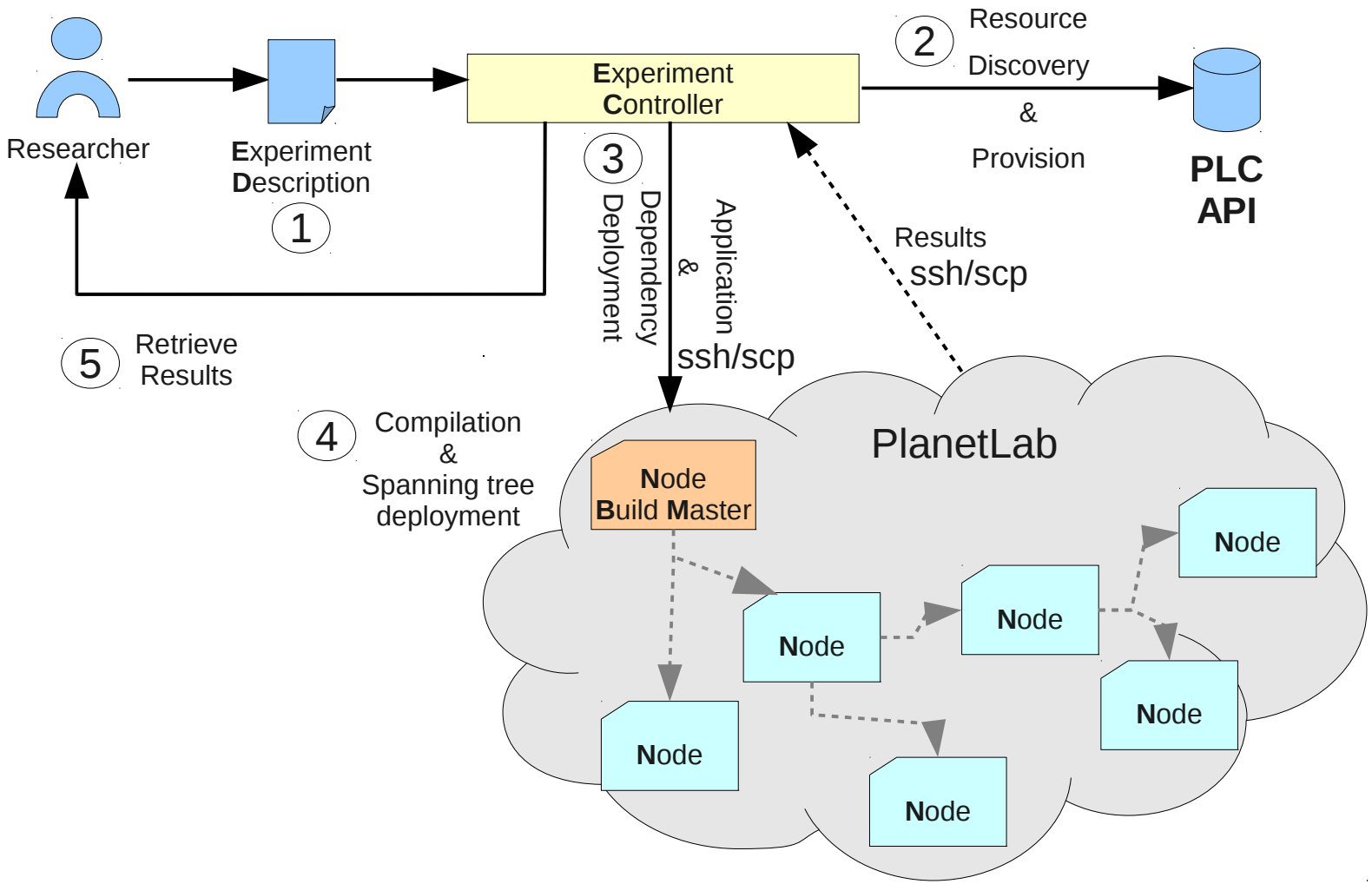
NEPI

Experiment description



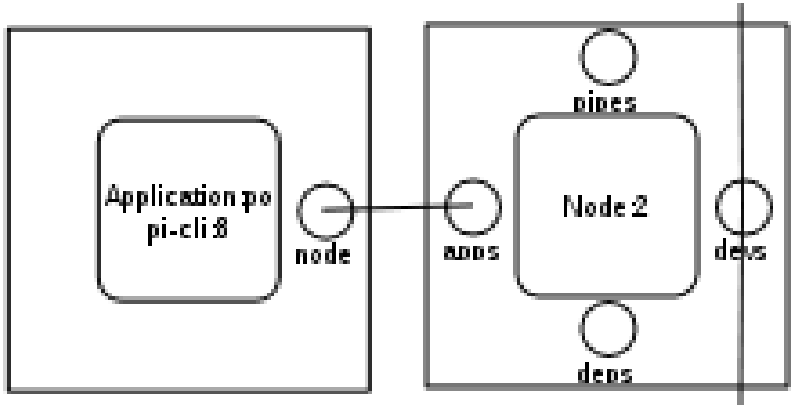
PlanetLab in NEPI

An overview



PlanetLab in NEPI

From design to deployment



Turns into

```
scp … <sources> <user>@node2:<somplace>
ssh … <user>@node2 <build commands>
ssh … <user>@node2 <launch commands>
```

PlanetLab in NEPI

From design to deployment

Routes become...

```
ssh … <user>@node2 sudo echo "<routes>"
                > /vsys/vroute.in
```

PlanetLab in NEPI

Tunnels

- Tunnels as special kinds of applications
  - **Layer 3** (Tun)
    - UDP, TCP, GRE links
  - **Layer 2** (Tap)
    - UDP, TCP, EGRE links

PlanetLab in NEPI

Routes

- **Vroute**
  - Safely manipulates the node's routing table
  - Enforces fair play among slices
  - No hard limit on number of users
  - Only preassigned IP ranges
- **Sliceip**: policy routing for PlanetLab
  - Highly flexible (Any IP range)
  - Can modify default routes
  - Limited scalability
- **NEPI chooses the best fit automatically**

The importance of node selection

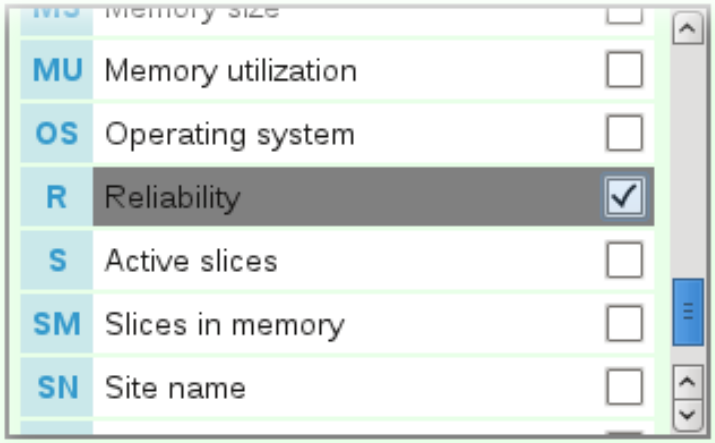
- **Node load can modify the outcome**
  - Overloaded nodes limit throughput
  - Underlay topology alters overlay characteristics
- **The real world isn't ideal**
  - Overloaded routers are real
    - It may be interesting to deploy some part of the experiment in overloaded nodes
  - The underlay can induce prioritization
    - Deep packet inspection
    - Pattern recognition

The importance of node selection

- **Node selection defines deployment success rates**
  - Unreachable nodes
  - Unreliable nodes
  - NAT'd nodes
  - Broken nodes
    - PlanetLab has many of those

The importance of node selection

- **Node monitoring is key**
  - **CoMon** provides useful metrics
  - PLC XML-RPC integrates well with it
  - NEPI leverages both



The importance of node selection

- **Node monitoring is key**
  - **CoMon** provides useful metrics
  - PLC XML-RPC integrates well with it
  - NEPI leverages both

ID	HOSTNAME	AU	ST	RES	BU	CF	L	R	?
15206	[illegible]	PLE	boot		63.7	0	1.7	100	
15207	[illegible]	PLE	boot		n/a	n/a	n/a	0	
15083	[illegible]	PLC	boot		445.2	97	0.8	100	
15085	[illegible]	PLC	boot		641	99	0.4	100	
353	[illegible]	PLC	boot		409.5	71	2.1	100	
14391	[illegible]	PLC	boot		656.2	93	1.3	100	
15003	[illegible]	PLE	boot...		188.7	99	0.1	65	
15022	[illegible]	PLE	boot		n/a	n/a	n/a	0	

Tunnel implementation

- GRE
- UDP
- TCP

Tunnel implementation

- **GRE**
  - Preserves underlay characteristics
  - Maximum performance
    - But zero flexibility
    - No obfuscation support
  - Trellis simplified
    - No classification
    - No bridges or switches
    - Just a point-to-point link
  - Safe and fool-proof isolation
    - GRE keys generated from slice ids
    - Supports many slices on the same link
  - Simple configuration with vsys

Tunnel implementation

- **UDP**
  - Good performance
  - Preserves underlay characteristics
    - Except prioritization if obfuscation is used
  - Allows extensive customization
    - Custom queues in Python or any other language
    - Custom aggregation methods in the form of stream filters
    - Or any other transformation one could think of
  - Requires explicit bandwidth limits

Tunnel implementation

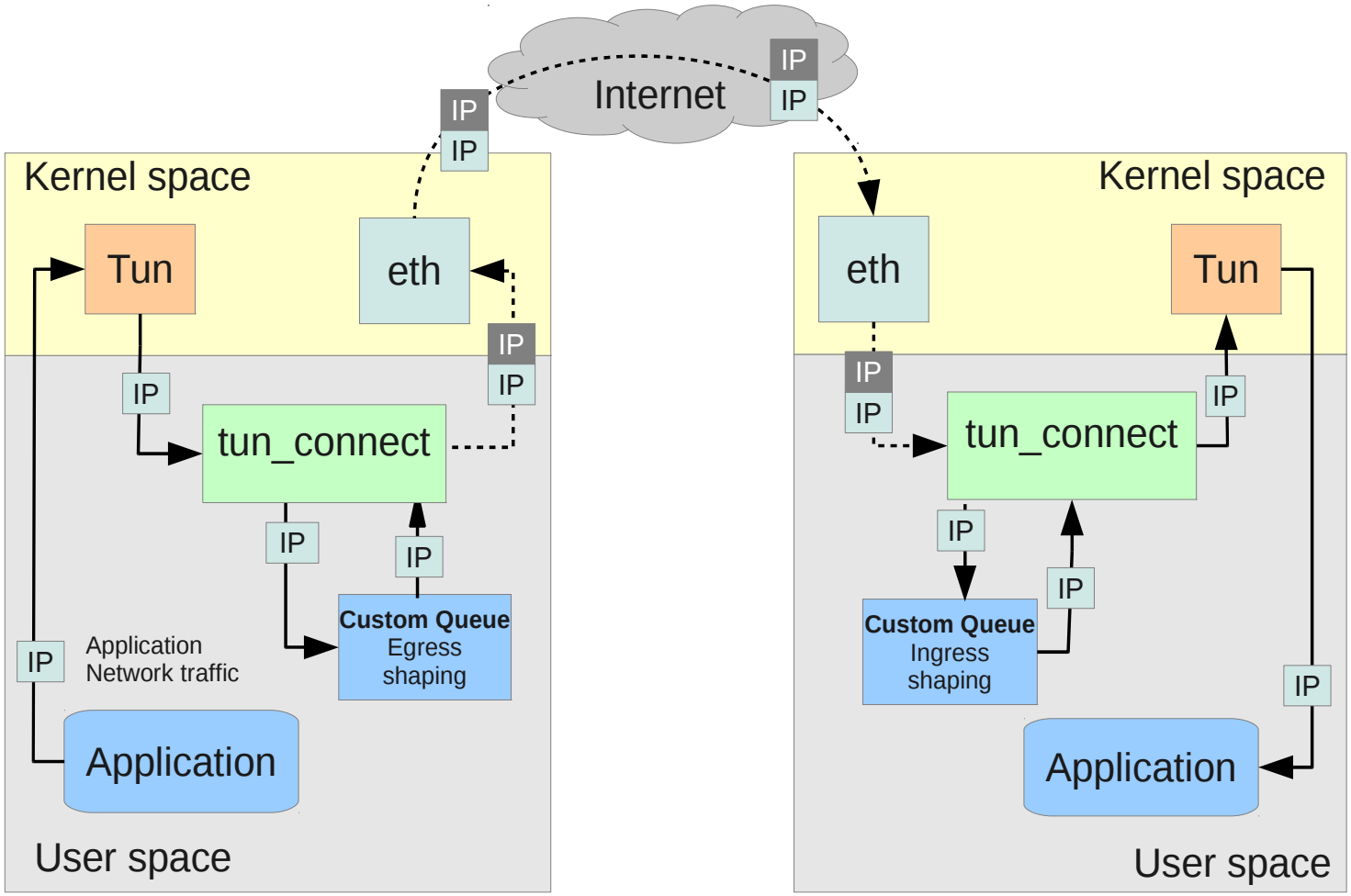
- **UDP**
  - Good performance
    - Lightweight compared to RiaS and PL-VINI
    - The kernel does the forwarding
      - Because we can manipulate routing tables
    - User mode only encapsulates packets
    - 200Mb/s with AES encryption
      - In preliminary tests
    - 1Gb/s without encryption

Tunnel implementation

- **TCP**
  - Lowest throughput of all methods
  - Hides underlay characteristics
  - Allows extensive customization
    - Like UDP
  - Traverses through NATs
    - Good to connect nodes on broadband or UMTS

Tunnel customization

Stream filters



Tunnel customization

What NEPI can do ...

- **Aggregation**
  - OverQoS
- **Queues**
  - New AQM schemes
  - Forcing specific test conditions
- **Instrumentation**
  - Traffic analysis and reporting
- **Traffic injection**

A NEPI use case

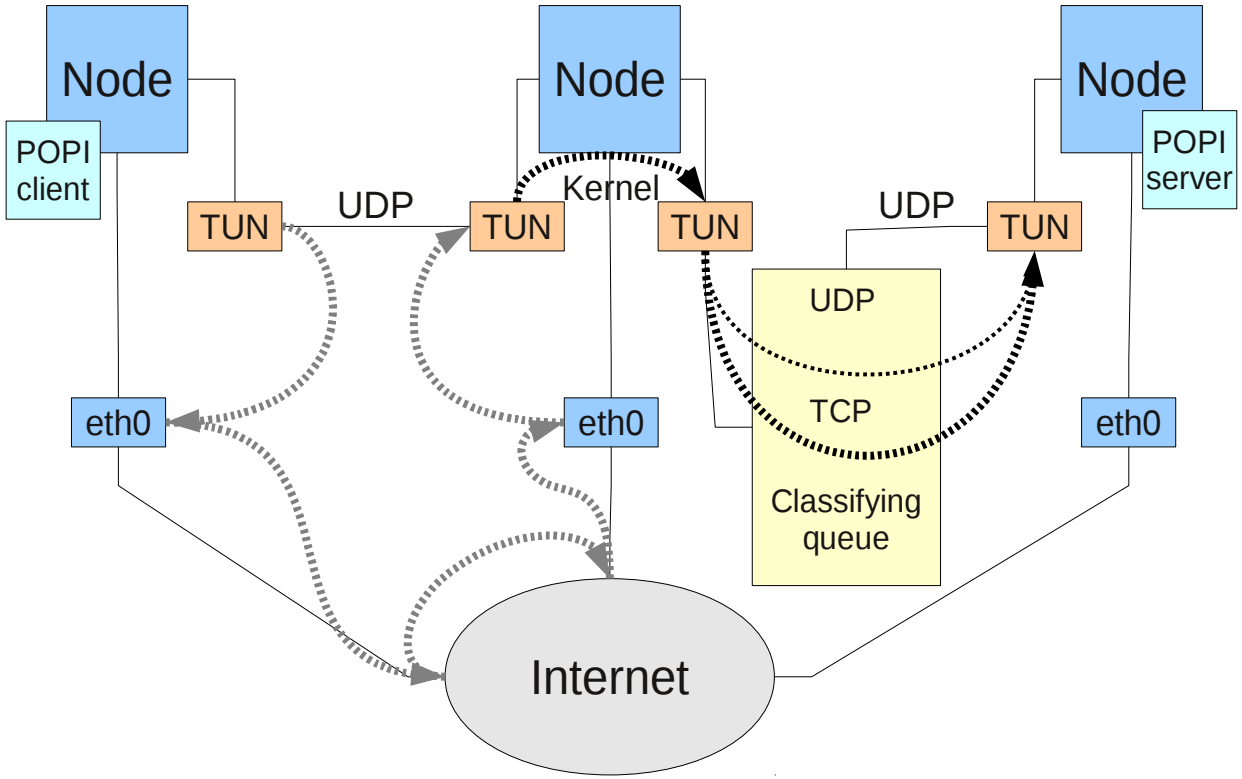
- A published experiment
  - **POPI,** Packet fOrwarding Priority Inference [1]
  - **Objective**
    - To infer traffic priority on the path between two endpoints
    - Tested in PlanetLab nodes
  - **Problem**
    - Experimental results could not be accurately verified
    - ISPs would provide incomplete information or not at all
    - No means to *know* if obtained results were reliable

[1] Lu, G., Chen, Y., Birrer, S., Bustamante, F.E., Li, X.: POPI: a user-level tool for inferring router packet forwarding priority. IEEE/ACM Trans. Netw. 18, 1–14 (2010)

POPI experiment with NEPI

- With NEPI we **can validate** POPI's results
  - Build an overlay
  - Use tunnel customization to force some characteristic
    - Like assigning 4x bandwidth to UDP
  - Run POPI on using the overlay
  - Compare results against what we know about the overlay
- A step in-between emulation and real-life tests
  - We control some parts of the environment
  - While still exposing the experiments to realistic traffic conditions
- In doing so, we test NEPI's effectiveness and weaknesses

POPI experiment with NEPI



POPI experiment with NEPI

Results

Runs/sets	Good	Bad	Fail
PlanetLab Europe	142/30	8/0	16/0
Dedicated Cluster	172/35	3/0	16/1

- Through automation, we tested in two environments:
  - PLE: real background traffic and load
  - A private PL-like cluster: devoid of extraneous activity
- Background noise does change results
  - Higher failure rates in PLE
  - Different results
    - POPI was less sensitive in PLE

POPI experiment with NEPI

Results

Runs/sets	Good	Bad	Fail
PlanetLab Europe	142/30	8/0	16/0
Dedicated Cluster	172/35	3/0	16/1

- 325 runs in 179 hours
  - Proves the value of automation
    - Through extensive automated testing, found POPI weak spots the original paper didn't find.
  - Would be prohibitive to do it manually

POPI experiment with NEPI

NEPI performance

- By using NEPI to validate POPI, we **did**
  - Prove NEPI is effective at assisting research
    - We were able to validate POPI more accurately than the original researchers
    - With comparable effort
  - Prove automation enables us to do more
    - We went beyond the original paper and tested POPI in multiple situations
    - We gathered extensive statistics
  - Prove that it is relevant to real cases
    - **POPI** is a real case
    - **OverQoS** is another real case (that would be possible)

POPI experiment with NEPI

NEPI performance

- By using NEPI to validate POPI, we **didn't**
  - Verify our techniques' theorized scalability
  - Prove it couldn't be done with other tools
    - It could have been done in PL-VINI
    - At a *much* higher cost
  - Quantitatively measure the amount of work required

Thank you

http://nepi.inria.fr